Available online at www.sciencedirect.com

ScienceDirect

Procedia Engineering 128 (2015) 15 – 24

Procedia Engineering

www.elsevier.com/locate/procedia

3rd European STAMP Workshop, STAMP EU 2015

Tool qualification considerations for tools supporting STPA

Sven Stefan Krauss[a,*], Martin Rejzek[a], Christian Hilbes[a]

[a]*ZHAW Zurich University of Applied Sciences, 8401Winterthur, Switzerland*

Abstract

We evaluated tool qualification requirements for hazard and risk analysis software tools, particularly for tools supporting System-Theoretic Process Analysis (STPA), and compared the tool qualification approaches of safety standards IEC 61508, EN 50128, DO-178C/DO-330 and ISO 26262. Our software tool SAHRA integrates STPA in an existing engineering toolchain by providing an extension for the UML/SysML modeling tool Sparx Systems Enterprise Architect. We found that the qualification of this tool according to the mentioned safety standards was not straightforward and required further analysis. Therefore, we analyzed the tool risks and found that those depend on many factors such as process risks, risks from tool errors, tool integration risks and operational scenarios regarding the use of the tool in the development lifecycle. We selected four operational scenarios for tools supporting STPA to evaluate tool qualification requirements. After concluding that a tool qualification is required, we used a multi-domain tool qualification development lifecycle guided by DO-330 for SAHRA.

© 2015 The Authors. Published by Elsevier Ltd. This is an open access article under the CC BY-NC-ND license (http://creativecommons.org/licenses/by-nc-nd/4.0/).
Peer-review under responsibility of the organizing committee of STAMP EU 2015

Keywords: STPA; STAMP; SAHRA; Multi Domain Tool Qualification; Tool Validation; Toolchain Analysis; Offline Support Tools; Cyber Physical Systems; Dependable Software

1. Introduction

Systems-Theoretic Process Analysis (STPA) is a modern hazard analysis technique developed by Leveson [1] and based on the Systems-Theoretic Accident Model and Processes (STAMP) accident model [2]. A hazard analysis with STPA begins with the definition of system accidents, hazards, high-level safety constraints and the construction of a hierarchical control structure, which shows how controllers and controlled processes interact via control actions and

* Corresponding author. Tel.: +41 58 934 47 87; fax: +41 58935 47 87.
E-mail address: svenstefan.krauss@zhaw.ch

1877-7058 © 2015 The Authors. Published by Elsevier Ltd. This is an open access article under the CC BY-NC-ND license (http://creativecommons.org/licenses/by-nc-nd/4.0/).
Peer-review under responsibility of the organizing committee of STAMP EU 2015
doi:10.1016/j.proeng.2015.11.500

feedback mechanisms. Based on the hierarchical control structure, unsafe control actions are identified (STPA Step 1). This information can be used to specify detailed safety constraints and safety requirements. For each unsafe control action, the causal scenarios are determined (STPA Step 2): how and what could cause the unsafe control action, or how and what causes the control action not being followed or executed properly. This information can then be used to identify controls and mitigations for the accident scenarios [1].

There are a number of software tools that support hazard analysis with STPA: A-STPA [3] and its successor XSTAMPP [4], based on Eclipse and developed as open source tools at the University of Stuttgart, Germany; SafetyHAT, a tool based on Microsoft Access and developed at the Volpe, The National Transportation Systems Center, USA [5]; an Eclipse-based STPA tool under development at MIT, USA [6]; SAHRA [7], a development of the Safety-Critical Research Lab of Zurich University of Applied Sciences (ZHAW), described later in this paper.

In cooperation with an industrial partner, the Safety-Critical Systems Research Lab of ZHAW evaluates the integration of STPA into our partner's engineering lifecycle in order to promote safety-guided design. Our industrial partner is a system components supplier for industrial, railway and defense applications. The main objective of our applied research project is the integration of STPA as a safety analysis method in our partner's engineering development lifecycle, by a) performing an industrial case study to show how STPA can be applied for system components engineering in multiple safety domains (confidential), and b) the development and integration of an STPA tool in the partner's engineering toolchain based on the experience gained from the case study and previous STPA projects [8], [9] and [10].

In this paper we present the approach we followed to integrate STPA in our partner's toolchain, and discuss the risks that are related to the integration of STPA. We introduce our software tool SAHRA and discuss risks related to using tools for STPA. For this task, we identified 4 operational scenarios for tools supporting STPA and evaluated the tool qualification requirements in multiple safety domains. An overview of this evaluation is given later in the paper. Finally, we present the tool qualification approach for our software tool SAHRA.

Nomenclature

EA	Enterprise Architect
SAHRA	STPA based Hazard and Risk Analysis
STAMP	System-Theoretic Accident Model and Processes
STPA	System-Theoretic Process Analysis
SysML	System Modelling Language
TCL	Tool Confidence Level
TD	Tool Error Detection
TI	Tool Impact
TQL	Tool Qualification Level
UML	Unified Modelling Language

2. An integrated STPA tool: SAHRA (STPA based Hazard and Risk Analysis)

In order to enable a safety-guided design, we suggest that requirements engineering and system, hardware and software engineering should be tightly coupled with STPA (see Fig. 1). Safety constraints and safety requirements must be provided to requirements engineering in order to specify detailed safety requirements and manage trace data to the analysis results. Requirements and trace data are the keys for ensuring safe design, verification, validation and modification. Requirements, safety constraints, risk control measures, and risk mitigation measures must be provided to system, hardware or software engineering in order to take appropriate design decisions, to create new designs or to modify existing designs until there are no unacceptable risks left [1].

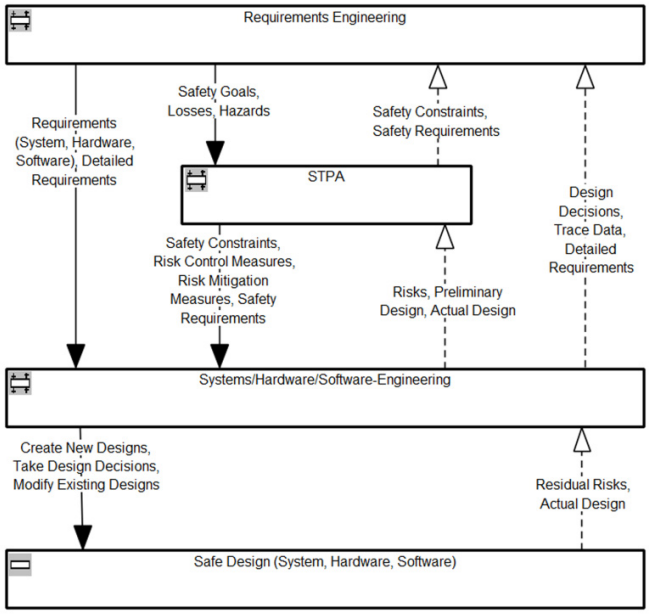

Fig. 1. Tightly coupled requirement driven design and safety-guided design visualized as hierarchical control structure.

The main risks for the development of safety-critical systems are: (L1) safety constraints and safety requirements are not provided or provided too late either to the system, hardware and software engineering when needed for design or to requirements engineering when needed for requirements specification; (L2) risk control measures and risk mitigation measures are not provided or provided too late to the system, hardware and software engineering when needed to make design decisions; (L3) trace data is incomplete or incorrect when performing change impact analysis for modifications.

To minimize the possibility of introducing human errors, engineering tools should be integrated. Tools are integrated if they work co-operatively, such that the outputs from one tool have a suitable content and format for automatic input to a subsequent tool, thus minimizing the possibility of introducing human error in the reworking of intermediate results [11].

For the integration of an STPA tool into our partner's engineering toolchain, we had to consider a number of constraints: (1) The STPA tool should be integrated into the existing toolchain, which is mainly based on the commercial UML/SysML modeling tool Enterprise Architect (EA) from Sparx Systems [12] used for requirements engineering and for system and software design; (2) Manual data export and import with external tools should be avoided; (3) tool qualification requirements should be considered when the tool is used for the development of safety-critical system components in industrial, railway, aerospace and defense projects. Under the aforementioned constraints, we decided to develop an extension called SAHRA (STPA based Hazard and Risk Analysis) for EA to integrate the STPA method directly in an UML/SysML environment (Fig. 2).

The corporate edition of EA already provides multi-user support with a security permission system, scripting and automation API, SQL searches, configuration management integration, report generation, and modeling functionality. Our motivation for this approach was that the STPA data is available in the same model repository with requirements and design data that allows full end-to-end traceability with requirements and design elements.

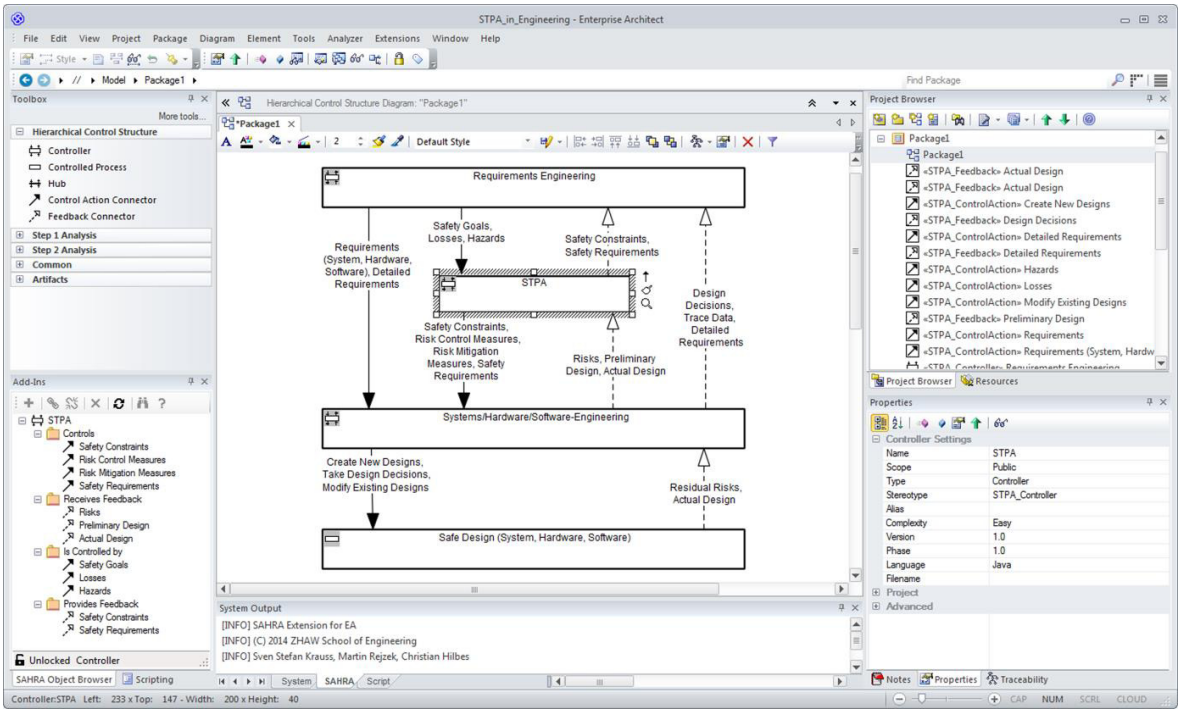

Fig. 2. Screenshot of the STPA extension SAHRA (STPA based Hazard and Risk Analysis) for Sparx Systems Enterprise Architect.

SAHRA comes with a special MDG (Model Driven Generation) profile for STPA based on EA's MDG technology [13], which provides additional toolboxes, UML profiles, patterns and templates for STPA modeling [14]. The extension provides a context-sensitive object browser and editor for STPA, diagram elements and special editors for performing STPA Step 1 and Step 2. The top level architecture of SAHRA is shown in Fig. 3.

With this approach, we fulfilled the previously mentioned constraints (1) and (2). To fulfill constraint (3), we reviewed 4 safety standards to see whether there was any applicable tool qualification requirements, and analyzed their potential impact for use and development of software tools supporting STPA: IEC 61508 [15] as a basic safety standard that is applicable to all kind of industry, EN 50128 [16] for railway, DO-178C [17] including DO-330 [18] for aerospace and defense, and ISO 26262 [19] for the automotive industry. While the first 3 standards are directly relevant for the products of our industrial partner, we also included ISO 26262 to review the state of the art of automotive tool qualification. The result of this analysis is given later in this paper.

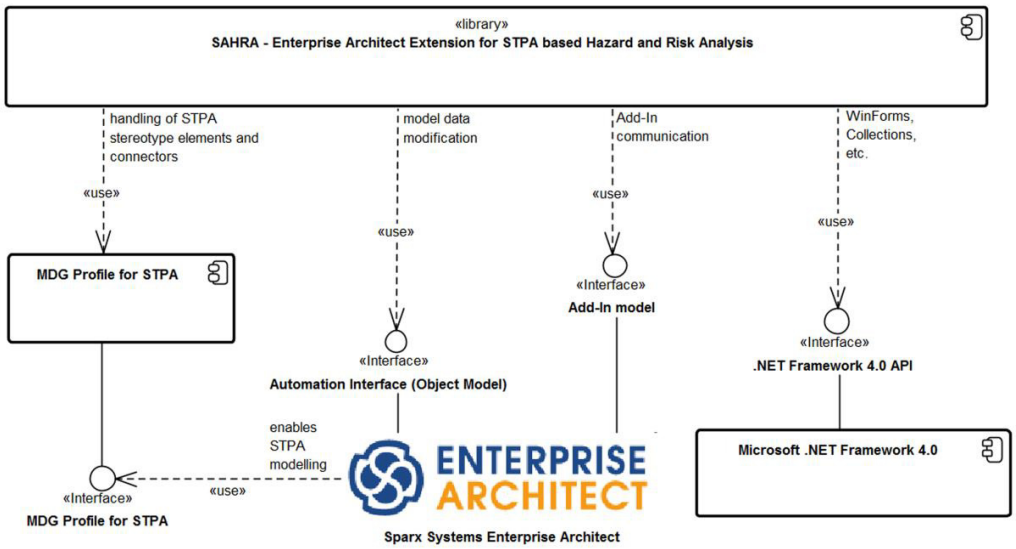

Fig. 3. Top level architecture of SAHRA (STBA based Hazard and Risk Analysis) realized as extension for Sparx Systems Enterprise Architect to integrate STPA methodology in the engineering toolchain.

3. Tool Qualification Considerations

3.1. Overview

Several safety standards require an assessment of whether an engineering tool might negatively impact safety, as malfunctioning engineering tools can introduce errors or fail to detect errors in the final safety-related system. Depending on this classification, a tool qualification may be required. In section J of a report by Hayhurst et al. [20] about tool qualification according to DO-178B [21], it is stated that a significant number of survey participants who qualified their tools found errors during tool qualification; 44% of the participants found errors in development tools and 57% found errors in verification tools. The tool errors would not have been detected if the users had not qualified their tools. There may be safety risks caused by errors in software tools that can be mitigated by tool qualification.

In the succeeding sections, we provide a brief overview of tool classification and tool qualification methods. Camus et al. [22] provide more information about this topic.

3.2. Basic safety standard IEC 61508

The requirements regarding tools are listed in Part 3, Chapter 7.4.4 of IEC 61508 [24]. The tool classes are defined in part 4 of IEC 61508 [23], section 3.2.11 to 3.2.13. Table 1 shows the definition and examples for the tool classes T1 to T3.

For each tool, the user must perform a tool classification to decide the tool class. Depending on the tool class, the tool qualification requirements differ. Tools of class T1 do not require further tool qualification measures. Tools of classes T2 and T3 require a documentation or specification, which specifies the tool behavior including instructions for its use, and requires a risk assessment. Tools of class T3 require tool validation, and for the ones of class T2 tool validation is recommended. As an alternative method to tool validation, tool qualification can be based on increased confidence from use.

Table 1. Tool classes as defined in Part 4 of IEC 61508 [23].

Class	Definition	Examples	Ref.
T1	Tool that generates no outputs that can directly or indirectly contribute to the executable code (including data) of the safety related system	Text Editor Requirements Management Tool Modeling Tool without Code Generation Configuration Management Tool	3.2.11
T2	Tool that supports the test or verification of the design or executable code, where errors in the tool can fail to reveal defects but cannot directly create errors in the executable software.	Test Generator Code Coverage Tool Static Code Analysis Tool	3.2.12
T3	Tool that generates outputs that can directly or indirectly contribute to the executable code of the safety related system	Optimizing Compiler Compiler with Runtime Package	3.2.1

3.3. Railway EN 50128

The railway tool qualification requirements are listed in Chapter 6.7 of EN 50128 [16]. The tool classes are defined in Chapter 3.1.42 to 3.1.43 (Table 2). The tool classes, tool classification and tool qualification requirements are very similar to those listed in the basic safety standard IEC 61508 [23]. The main differences are that more tool qualification methods and examples are listed. The most important tool qualification methods are: tool validation, confidence from use, and other independent error detection means such as diverse redundant code that has the potential to detect tool errors as well.

Table 2. Tool classes as defined in EN 50128 [16].

Class	Definition	Examples	Ref.
T1	Tool that generates no outputs that can directly or indirectly contribute to the executable code (including data) of the safety related system	Text Editor Requirements Management Tool Modeling Tool without Code Generation Configuration Management Tool	3.1.42
T2	Tool that supports the test or verification of the design or executable code, where errors in the tool can fail to reveal defects but cannot directly create errors in the executable software.	Test Generator Code Coverage Tool Static Code Analysis Tool	3.1.43
T3	Tool that generates outputs that can directly or indirectly contribute to the executable code of the safety related system	Optimizing compiler Compiler with runtime package Data/Algorithm compiler Tool for changing reference values during operation	3.1.44

3.4. Aerospace and Defense DO178C/DO-330

DO-330 is a supplement to the DO-178C [17] standard dedicated to tool qualification [18]. This supplement describes the tool qualification process from a user's perspective, and includes also a detailed safety tool development lifecycle for tool developers. The requirements for tool qualification depend on the tool qualification level (TQL), which is built from tool criteria, and the safety level of the software (see Table 3). There are several tables in DO-330 Annex A [18] that define for each tool qualification level what sections and objectives of the standard are applicable. TQL-5 is the lowest level that requires tool validation with requirements based testing where TQL-1 is the highest level that demands a compliant tool development process. Qualification for TQL-5 can be done by the tool user, where for other levels support of the tool developer is necessary to provide the required tool qualification data.

Table 3. Tool criteria and tool qualification level (TQL) per software level according to DO-330 [18].

Tool Criteria	Definition	Level D	Level C	Level B	Level A
1	A tool whose output is part of the airborne software and thus could insert an error.	TQL-4	TQL-3	TQL-2	TQL-1
2	A tool that automates verification process(es) and thus could fail to detect an error, and whose output is used to justify the elimination or reduction of: 1. Verification process(es) other than that automated by the tool, or 2. Development process(es) that could have an impact on the airborne software.	TQL-5	TQL-5	TQL-4	TQL-3
3	A tool that, within the scope of its intended use, could fail to detect an error.	TQL-5	TQL-5	TQL-5	TQL-4

3.5. Automotive ISO 26262

The automotive tool qualification requirements are listed in Part 8, Chapter 11 of ISO 26262 [19]. The approach for tool classification is rather different from the standards described above; the requirements for tool qualification depend on the tool confidence level (TCL), which is built from the tool impact (TI), and the confidence level whether an error caused by the tool could be detected or prevented, called tool error detection (TD) (see Table 4).

Table 4. Tool confidence level built from tool impact (TI) and tool error detection (TD) according to ISO 26262-8 [19]

Tool Impact		Tool Error Detection		
The possibility that a malfunction of a particular software tool can introduce or fail to detect errors in a safety-related item or element being developed.		High confidence to prevent/detect erroneous outputs	Medium confidence to prevent/detect erroneous outputs	Other cases
		TD1	TD2	TD3
TI1 shall be selected when there is an argument that there is no such possibility	TI1	TCL1	TCL1	TCL1
TI2 shall be selected in all other cases	TI2	TCL1	TCL2	TCL3

The listed tool qualification methods are: (1a) increased confidence from use, (1b) evaluation of the tool development process, (1c) validation of the software, and (1d) development according to a safety standard. Recommended or highly recommended methods of tool qualification are listed for each automotive safety integrity level (ASIL). For the highest safety level ASIL D in combination with tool confidence level TCL3 or TCL2, tool validation is highly recommended (i.e. method 1c); alternatively, it is highly recommended to use a tool that has been developed according a safety standard (i.e. method 1d).

There is a remark to method (1d) listed in Tables 4 and 5 in Part 8 of ISO 26262 [19]: “No safety standard is fully applicable to the development of software tools. Instead, a relevant subset of requirements of the safety standard can be selected.” At the time the standard was published no special tool development standard existed. In the meantime, the dedicated tool qualification standard DO-330 from the aerospace domain became available, which can also be used for the development of safety-related tools. DO-330 Section 1.2.c explicitly states “This document provides guidance for airborne and ground-based software. It may also be used by other domains, such as automotive, space, systems, electronic hardware, aeronautical databases, and safety assessment processes” [18].

4. Classification of tools supporting STPA

For a proper tool classification, not only the tool itself should be considered in the risk assessment, but also the way in which the tool is integrated and used in the whole toolchain [25]. Asplund [26] adapted STPA and applied it to tool integration. In his STPA-based analysis of toolchains, Asplund [27] identified 8 lacking properties as causes to risks: Lack of Data Integrity (P1), Lack of Traceability for Completeness and Consistency (P2), Lack of Automated Transformations of Data (P3), Lack of Formally Defined Data Semantics (P4), Lack of Possibility to Automate Tool

Usage (P5), Lack of Data Mining (P6), Lack of Process Notifications and Process Control (P7), and Lack of Possibility to Create Customized GUIs (P8).

For the preliminary risk assessment, we will use "Lack of Data Integrity" (P1) and "Lack of Traceability for Completeness and Consistency" (P2) as an example. A typical STPA tool manages STPA data and can be described as a "requirements management tool" or as a "modeling tool without code generation". When a tool is used for managing STPA data, the tool might impose the following risks: (R1) Analysis data (i.e. safety requirements, safety constraints, risk control and mitigation measures) is incomplete or corrupt (chapter 2 →L1, →L2); (R2) Trace data is incomplete or corrupt (chapter 2 →L3) (e.g., caused by an error in a document generator or the data export function or by an inadequately handled file system error in the save function). These tool errors may be difficult to detect, depending on the tool operational scenarios, the way the tool is used, and the functionality it provides.

We identified 4 operational scenarios and evaluated the requirements for tool qualification: 1) STPA data is managed in a standalone tool whose output is manually verified for completeness and consistency; 2) STPA data is automatically transferred to or integrated into another tool without manual verification; 3) STPA data is used for verification like formal model checking; 4) STPA data is used for auto code generation.

Operational Scenario 1: STPA data is managed in a standalone tool whose output is manually verified for correctness and completeness. For this scenario the applicable tool class is T1 for industrial and railway industries; for automotive industries, the applicable Tool Confidence Level is TCL1 (T1 for tool impact is selected) and for the aerospace industry, a tool qualification is not required. For all standards, no tool qualification is required, but the quality assurance process must ensure that no tool output is used without manual verification.

Operational Scenario 2: STPA data is automatically transferred to or integrated into another tool without manual verification (e.g., STPA extension to a modeling tool where data is stored in the model repository). The manual verification process for correctness and completeness of the tool output is eliminated or is not possible. For this scenario the applicable Tool Confidence Level in automotive is TCL1 to TCL2 depending on the tool's implementation of tool error detection mechanisms (TD1 or TD2) or TCL3 in other cases (TD3). For the aerospace industry, tool criteria 2 is selected (the verification process is skipped) and the applicable Tool Qualification Level is TQL-3 for level A and TQL-4 for level B to level D. There is no clear guidance in industry and railway standards for this case, which suggests the tool class would still be T1 for both domains (T2 would be recommended).

Operational Scenario 3: STPA data is used for verification (e.g., STPA tool includes a formal model checker). In this case the STPA tool is used as a verification tool, which might fail to detect an error. The tool class is: T2 for industrial and the railway; TCL1, TCL2 or TCL3 in the automotive industry, depending on error detection methods; TQL-4 or TQL-5 for the aerospace industry, depending on the safety level.

Operational Scenario 4: STPA data is used for auto code generation (e.g., auto code generator uses STPA data like a safety control loop and process model to generate code. In this case, the STPA tool is used as code generation tool, which might introduce an error in the final safety-related system if there is an error with STPA data. For this type, tool classes are: T3 for industrial and railway industries; TCL1, TCL2 or TCL3 in the automotive industry, depending on error detection methods; TQL-1, TQL-2 or TQL-3 for the aerospace industry, depending on the safety level. The 4 operational scenarios and respective classifications are summarized in Table 5.

Table 5. Classification of tools supporting STPA.

Operational Scenario	IEC61508	EN 50128	ISO 26262	DO-330
1	T1	T1	TCL1	---
2	(T1)|T2	(T1)|T2	TCL1|TCL2|TCL3	TQL-3|TQL-4
3	T2	T2	TCL1|TCL2|TCL3	TQL-4
4	T3	T3	TCL1|TCL2|TCL3	TQL-1|TQL-2|TQL-3

5. Conclusion

Most of the reviewed standards require tool qualification only for tools that support the software development lifecycle (offline support tools). One can argue that when safety analysis tools are used on system level, tool qualification requirements are not applicable. We propose that tool risk assessment and tool qualification should not

be limited to software lifecycle supporting tools, but should be performed for all engineering tools supporting the development of safety-related systems. In particular, safety analysis tools should be qualified, so tool users can rely on their correct functionality and outputs.

We also claim that a tool safety assessment should not only be based on the listed tool criteria and classes, because modern tools may have integrated functionality, or the functionality may be integrated in other tools. For example, if an STPA tool has a built-in formal model checker, the tool is considered to be a verification tool, which requires tool qualification. A systematic view on tool qualification in tool chains is therefore recommended [28].

Developers of safety-related software tools should consider whether they use DO-330 as tool development lifecycle in order to avoid systematic failures in the tool. This standard includes tool validation, which can be used for tool qualification in the other reviewed safety domains. Tool developers of STPA tools should consider whether they provide tool qualification packages for their tools, so that tool users can rely on the correct functionality.

The reviewed standards demand that tool users classify their software tools and perform a risk assessment that shows what impact tool error might have on the final safety-critical system. Typical tool qualification methods are: a) increased confidence from use by having evidence for an extensive history of satisfactory use; b) tool validation by requirements-based testing; c) tool development according to a safety standard.

SAHRA creates the STPA data in the model repository. In order to enable an efficient use of the tool, manual verification steps should be eliminated. This approach fits to the tool operational scenario 2, which requires tool qualification. New software tools like SAHRA do not have an extensive history of use and cannot be qualified by increased confidence from use. All standards allow tool validation by requirements-based testing as a tool qualification method. Users of standalone STPA tools have to ensure that they have a proper manual verification process in place that checks the completeness and correctness of the tool output.

With the software tool SAHRA [7], we were able to integrate STPA in a UML/SysML environment. We used a tool development lifecycle guided by DO-330 with TQL-4 as a target level as means to ensure the high quality of the integration of our software tool SAHRA and avoid systematic failures during development. We plan to provide a multi-domain tool qualification support package including tool operational requirements, pre-defined test cases and procedures, reference workflow, and a safety manual.

Acknowledgements

The project is funded by the Swiss Commission for Technology and Innovation (CTI), project grant number 15822.1 PFIW-IW, and by Curtiss-Wright Drive Technology GmbH, Switzerland.

References

[1] N.G. Leveson, Engineering A Safer World: Systems Thinking Applied to Safety, MIT Press, Cambridge, MA, 2011.
[2] N. Leveson, A new accident model for engineering safer systems, Safety Science. 42 (2004) 237–270.
[3] A. Abdulkhaleq, S. Wagner, A-STPA: Open Tool Support for System-Theoretic Process Analysis, STAMP Workshop 2014, MIT, Boston, 2014.
[4] A. Abdulkhaleq, S. Wagner, XSTAMPP: An eXtensible STAMP Platform As Tool Support for Safety Engineering, STAMP Workshop 2015, MIT, Boston, 2015.
[5] Q. Hommes, The Volpe STPA Tool, STAMP Workshop 2014, MIT, Boston, 2014.
[6] D. Suo, J. Thomas, An STPA Tool, STAMP Workshop 2014, MIT, Boston, 2014.
[7] Safety-Critical Systems Research Lab Team of ZHAW Zurich University of Applied Sciences, SAHRA - STPA based Hazard and Risk Analysis, http://www.sahra.ch (accessed 1 August 2015).
[8] B. Antoine, Systems Theoretic Hazard Analysis (STPA) Applied to The Risk Review of Complex Systems: An Example From The Medical Device Industry, PhD Thesis, MIT, Boston, 2013.
[9] M. Rejzek, Evaluation of STPA in the Safety Analysis of the Gantry 2 Proton Radiation Therapy System, STAMP Workshop 2012, MIT, Boston, 2012.
[10] M. Rejzek, Use of STPA in Digital Instrumentation and Control Systems of Nuclear Power Plants, 2nd European STAMP Workshop, Stuttgart, 2014.
[11] IEC, Overview of techniques and measures, in: IEC, Functional Safety of Electrical/Electronic/Programmable Electronic Safety-Related Systems 61508-7, second ed., International Electrotechnical Commission, Geneva, 2010.
[12] Sparx Systems Pty Ltd. Enterprise Architect, UML Modeling and Lifecycle Tool Suite. [Online]. Available: http://www.sparxsystems.com.

[13] Sparx Systems Pty Ltd., Model Driven Generation (MDG) Technologies. [Online]. Available: http://www.sparxsystems.com.au/resources/mdg_tech/
[14] M. Rejzek, C. Hilbes, SS. Krauss, Safety Driven Design with UML and STPA, STAMP Workshop 2015, MIT, Boston, 2015.
[15] IEC, Functional Safety of Electrical/Electronic/Programmable Electronic Safety-Related Systems 61508, second ed., International Electrotechnical Commission, Geneva, 2010.
[16] CENELEC, Railway Applications - Communication, Signalling and Processing Systems - Software for Railway Control and Protection Systems, Irish Standard EN 50128:2011&AC:2014, NSAI Standards, Dublin, 2014.
[17] RTCA, Software Considerations in Airborne Systems and Equipment Certification DO-178C, Radio Technical Commission for Aeronautics, Washington, 2011.
[18] RTCA, Software Tool Qualification Considerations DO-330, Radio Technical Commission for Aeronautics, Washington, 2011.
[19] ISO, Supporting processes, in: ISO, Road Vehicles - Functional Safety 26262-8:2011, International Standards Organization, Geneva, 2011.
[20] K.J. Hayhurst, C. A. Dorsey, J. C. Knight, N. G. Leveson, G. F. McCormick, Streamlining Software Aspects of Certification: Report on the SSAC Survey, NASA/TM-1999-209519, Hampton, Virginia,1999.
[21] RTCA, Software Considerations in Airborne Systems and Equipment Certification DO-178B Radio Technical Commission for Aeronautics, Washington, 1992.
[22] J-L Camus, M. P. Dewalt, F. Pothon, G. Ladier, J-L Boulanger, J-P Blanquart, Tool Qualification in Multiple Domains: Status and Perspectives, ERTS^2 Conference, Toulouse, France, 2014.
[23] IEC, Definitions and abbreviations, in: IEC, Functional Safety of Electrical/Electronic/Programmable Electronic Safety-Related Systems 61508-4, second ed., International Electrotechnical Commission, Geneva, 2010.
[24] IEC, Software requirements, in: IEC, Functional Safety of Electrical/Electronic/Programmable Electronic Safety-Related Systems 61508-3, second ed., International Electrotechnical Commission, Geneva, 2010.
[25] F. Asplund, M. Biehl, J. El-khoury, D. Frede, M. Törngren, Tool Integration: from Tool to Tool Chain with ISO 26262, SAE Technical Paper 2012-01-0026, SAE 2012 World Congress & Exhibition, Detroit, 2012.
[26] F. Asplund, Risks Related to the Use of Software Tools when Developing Cyber-Physical Systems, PhD Thesis, KTH Royal Institute of Technology, Stockholm, Sweden, 2014.
[27] F. Asplund, Safety and Tool Integration: A System-Theoretic Process Analysis, Technical Report ISRN/KTH/MMK-R-12/01-SE, KTH Royal Institute of Technology, Stockholm, Sweden, 2012.
[28] F. Asplund, J. El-khoury, M. Törngren, Qualifying software tools: a systems approach, in: F. Ortmeier, P. Daniel (Eds.), Computer Safety, Reliability, and Security, Springer Berlin Heidelberg, Berlin, 2012. pp. 340-351.